



ECP Software and Preparing for Heterogeneity

Michael Heroux, ECP Director of Software Technology
Sandia National Laboratories

ECP ST Overview

Scope

Deliver a software stack that enables sustainable exascale capabilities

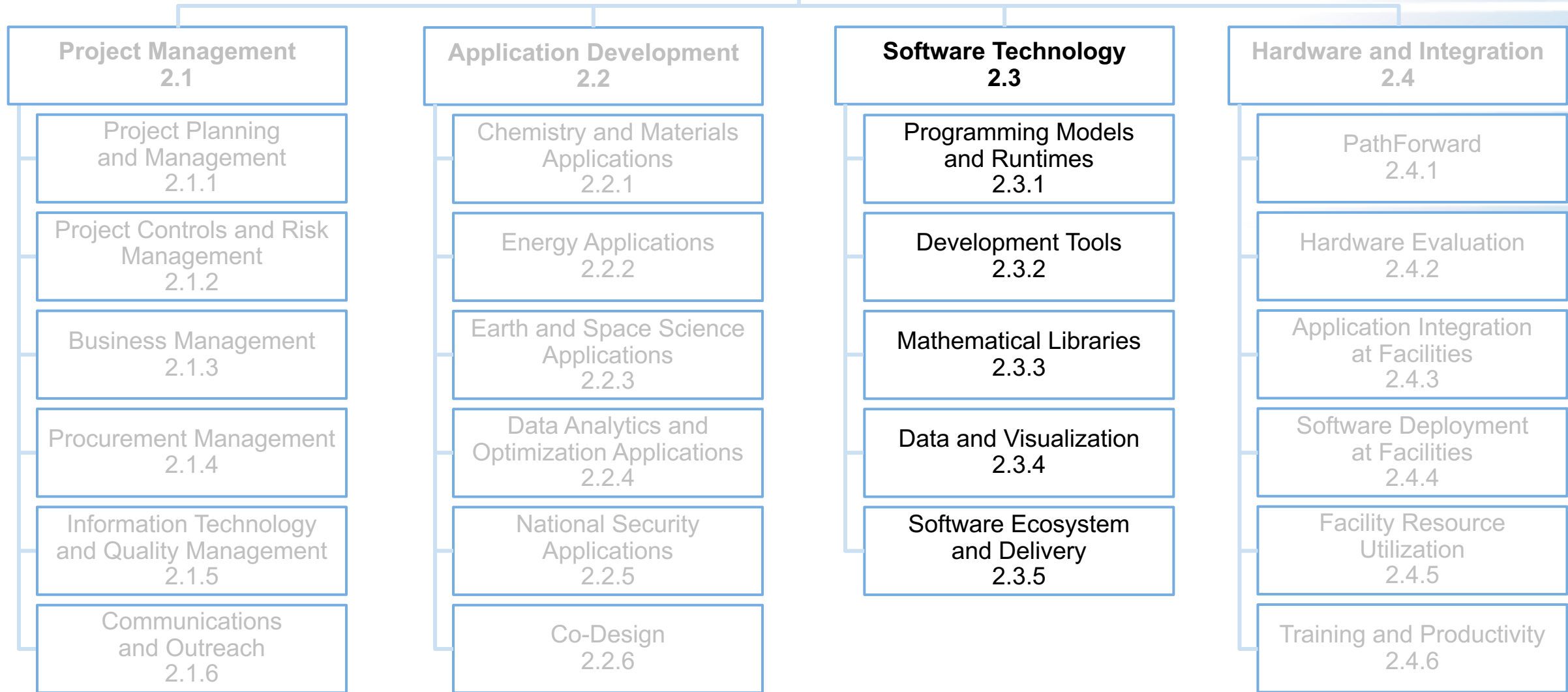
Mission
need

Objective

Capabilities across the entire HPC software stack that **complement** and **coordinate** with **facilities**, **vendors** and other software providers to enable effective execution of **ECP apps**, and deliver a capable, sustainable **exascale ecosystem**.

Provide the **next generation** of DOE software capabilities targeted toward exascale applications and platforms. Provide these capabilities for the **specific exascale systems** as a high quality, sustainable product suite.

Exascale Computing Project 2.0



ECP Software Technology Leadership Team

**Mike Heroux, [Software Technology](#) Director**

Mike has been involved in scientific software R&D for 30 years. His first 10 were at Cray in the LIBSCI and scalable apps groups. At Sandia he started the Trilinos and Mantevo projects, is author of the HPCG benchmark for TOP500, and leads productivity and sustainability efforts for DOE.

**Jonathan Carter, [Software Technology](#) Deputy Director**

Jonathan has been involved in the support and development of HPC applications for chemistry, the procurement of HPC systems, and the evaluation of novel computing hardware for over 25 years. He currently a senior manager in Computing Sciences at Berkeley Lab.

**Rajeev Thakur, [Programming Models and Runtimes](#) (2.3.1)**

Rajeev is a senior computer scientist at ANL and most recently led the ECP Software Technology focus area. His research interests are in parallel programming models, runtime systems, communication libraries, and scalable parallel I/O. He has been involved in the development of open source software for large-scale HPC systems for over 20 years.

**Jeff Vetter, [Development Tools](#) (2.3.2)**

Jeff is a computer scientist at ORNL, where he leads the Future Technologies Group. He has been involved in research and development of architectures and software for emerging technologies, such as heterogeneous computing and nonvolatile memory, for HPC for over 15 years.

**Lois Curfman McInnes, [Math Libraries](#) (2.3.3)**

Lois is a senior computational scientist in the Mathematics and Computer Science Division of ANL. She has over 20 years of experience in high-performance numerical software, including development of PETSc and leadership of multi-institutional work toward sustainable scientific software ecosystems.

**Jim Ahrens, [Data and Visualization](#) (2.3.4)**

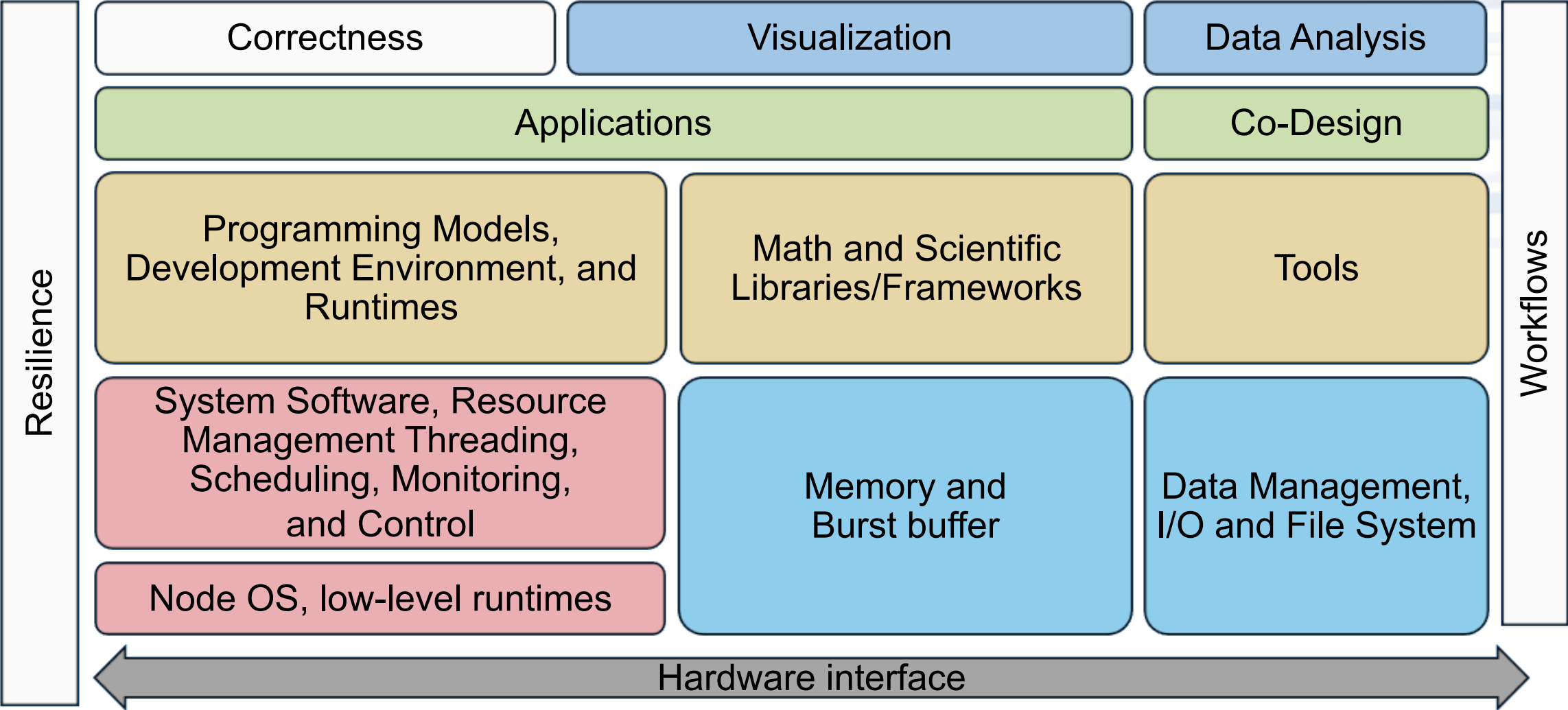
Jim is a senior research scientist at the Los Alamos National Laboratory (LANL) and an expert in data science at scale. He started and actively contributes to many open-source data science packages including ParaView and Cinema.

**Rob Neely, [Software Ecosystem and Delivery](#) (2.3.5)**

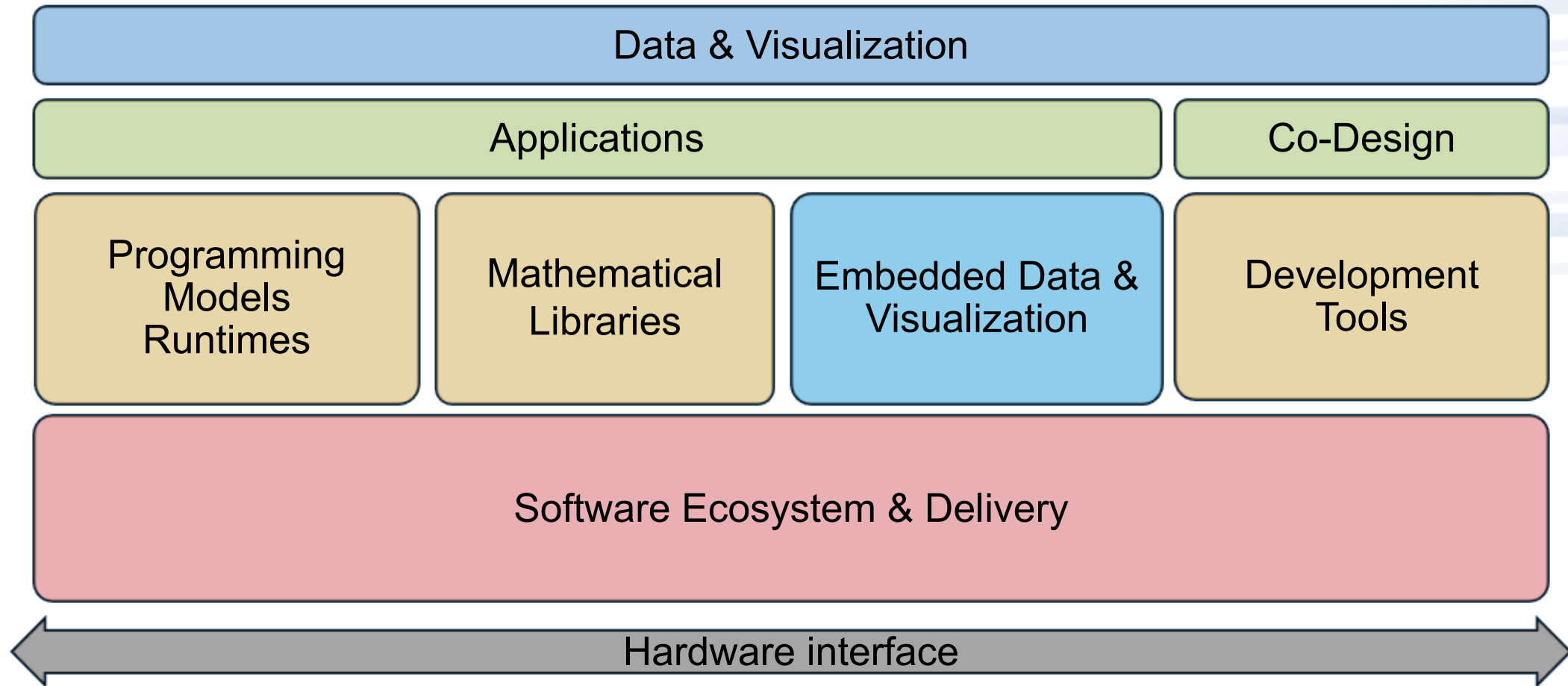
Rob has several leadership roles at LLNL spanning applications, CS research, platforms, and vendor interactions. He is an Associate Division Leader in the Center for Applied Scientific Computing (CASC), chair of the Weapons Simulation and Computing Research Council, and the lead for the Sierra Center of Excellence.



ECP ST SW Stack Version 1



ECP ST SW Stack Version 2



ECP software: Productive, sustainable ecosystem

Goal

Build a comprehensive, coherent software stack that enables application developers to productively write highly parallel applications that effectively target diverse exascale architectures

Extend current technologies to exascale where possible



Perform R&D required for new approaches when necessary



Coordinate with and complement vendor efforts



Develop and deploy high-quality and robust software products



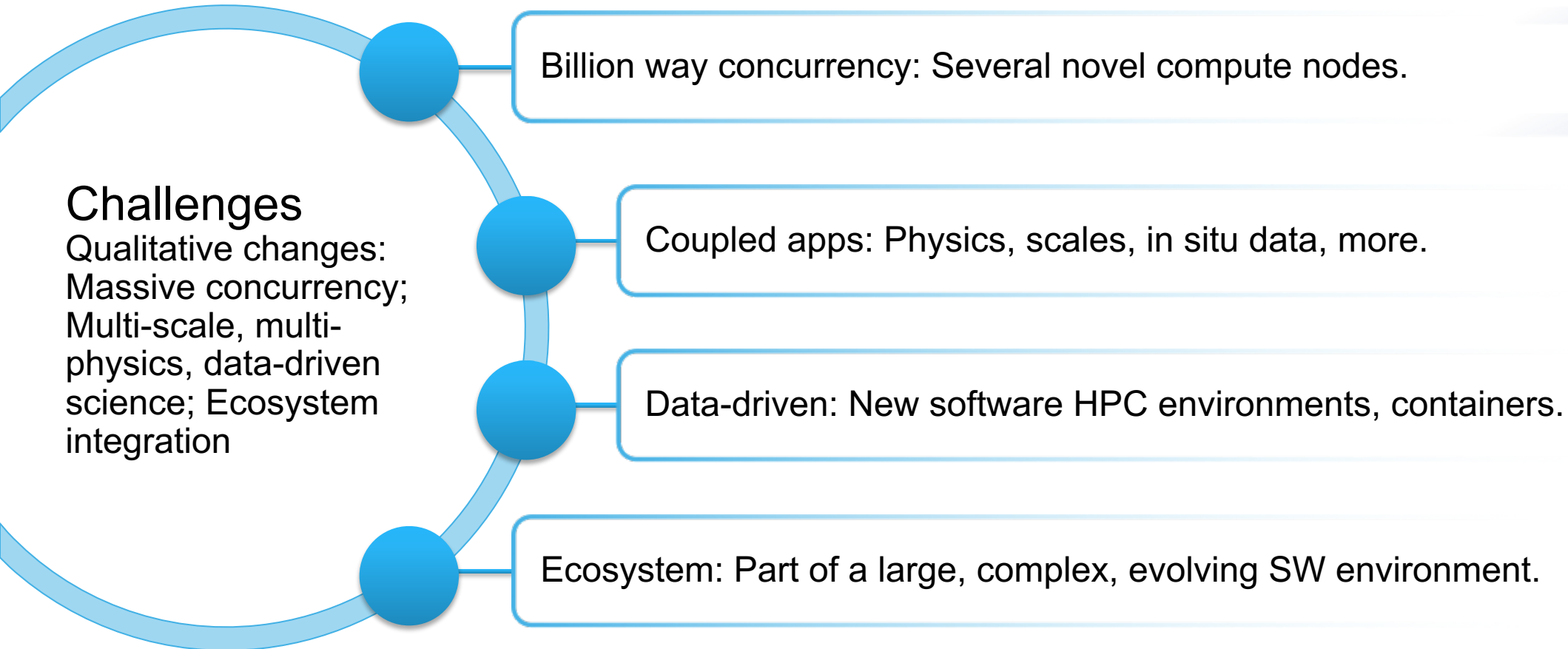
56 WBS L4 subprojects executing RD&D

195 L4 subproject (P6) milestones delivered in FY17

426 L4 subproject (P6) milestones planned in FY18-19

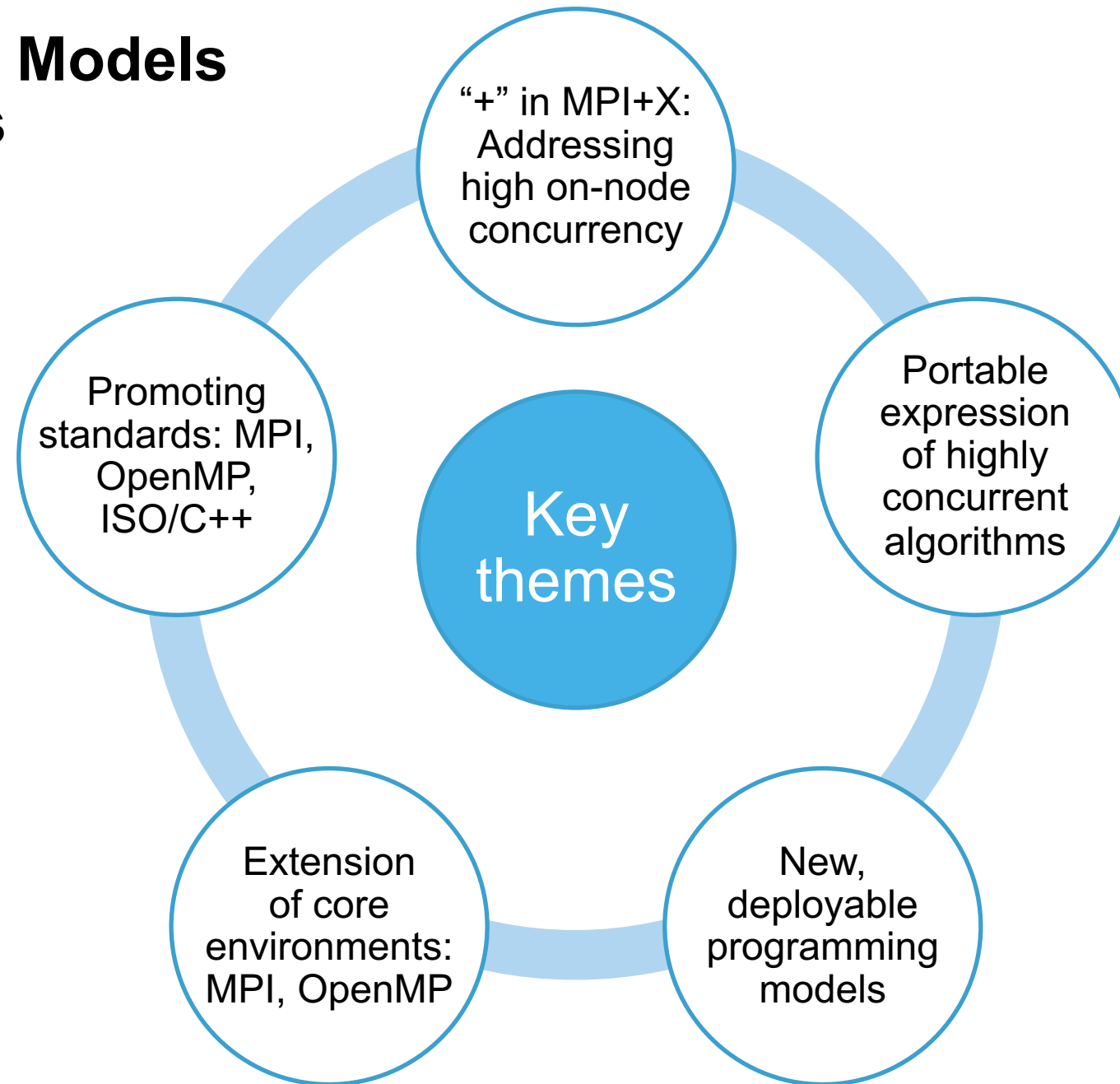


ECP software: Challenges

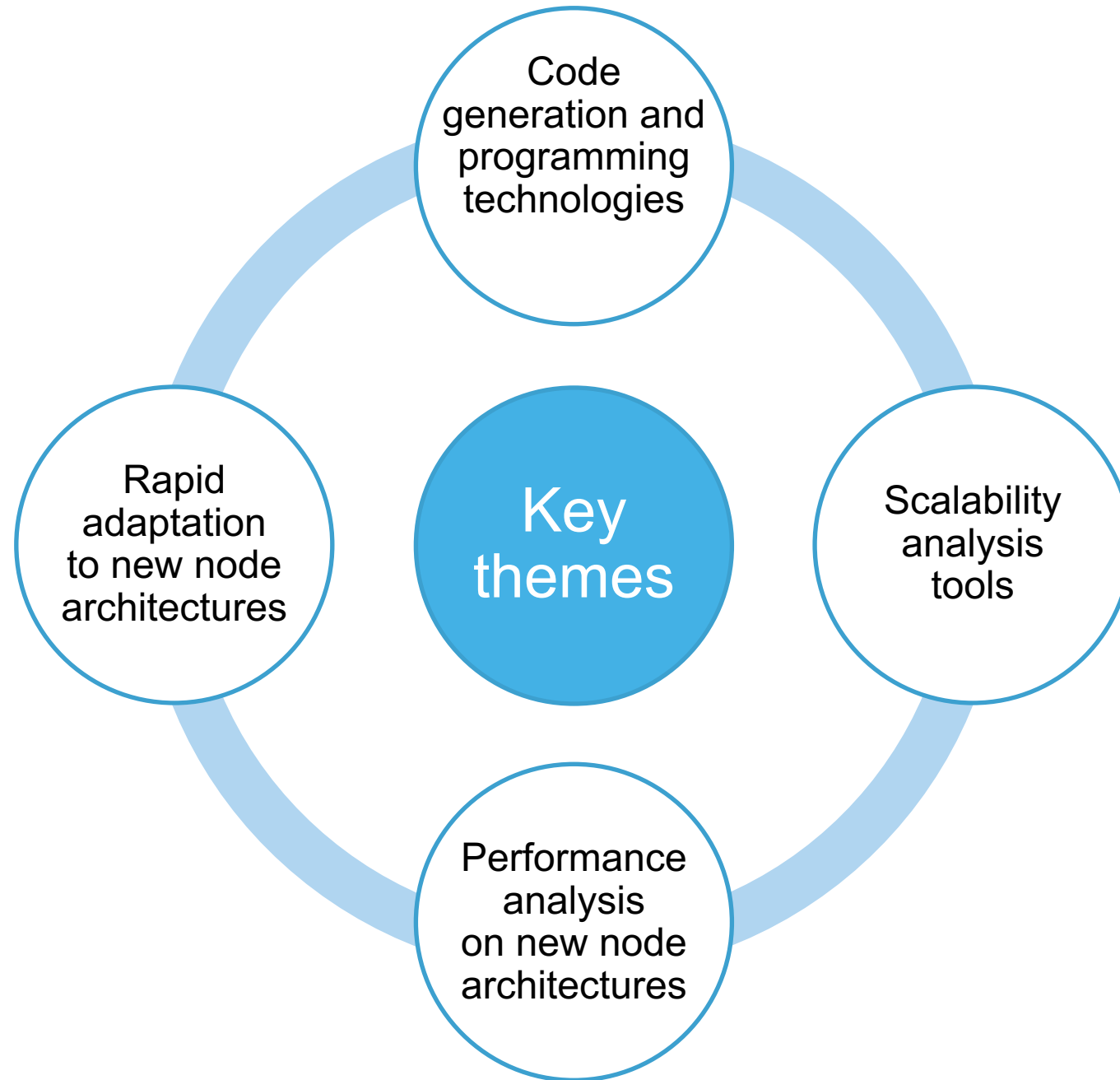


Overview of ECP ST Technical Areas

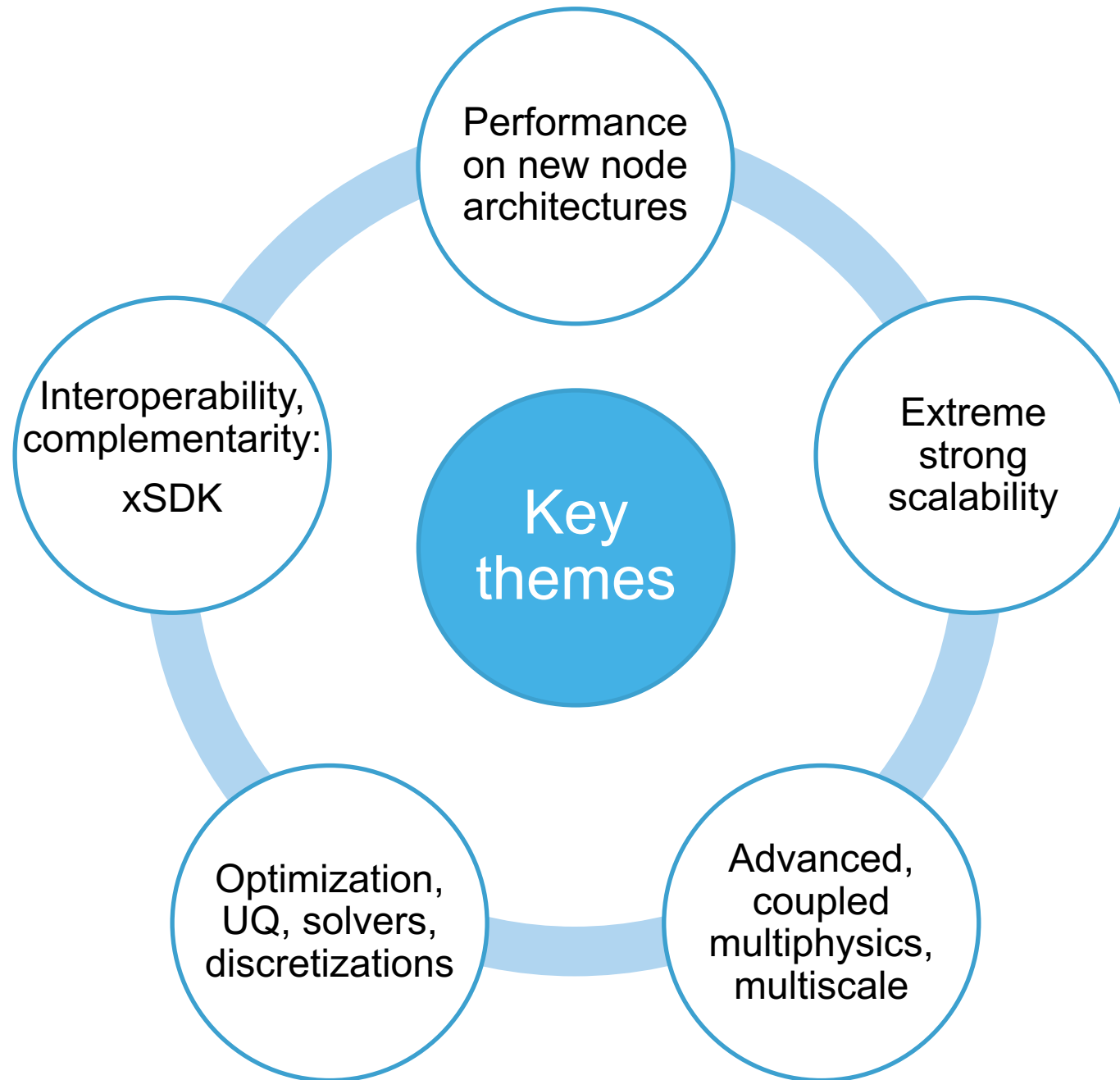
Programming Models and Runtimes



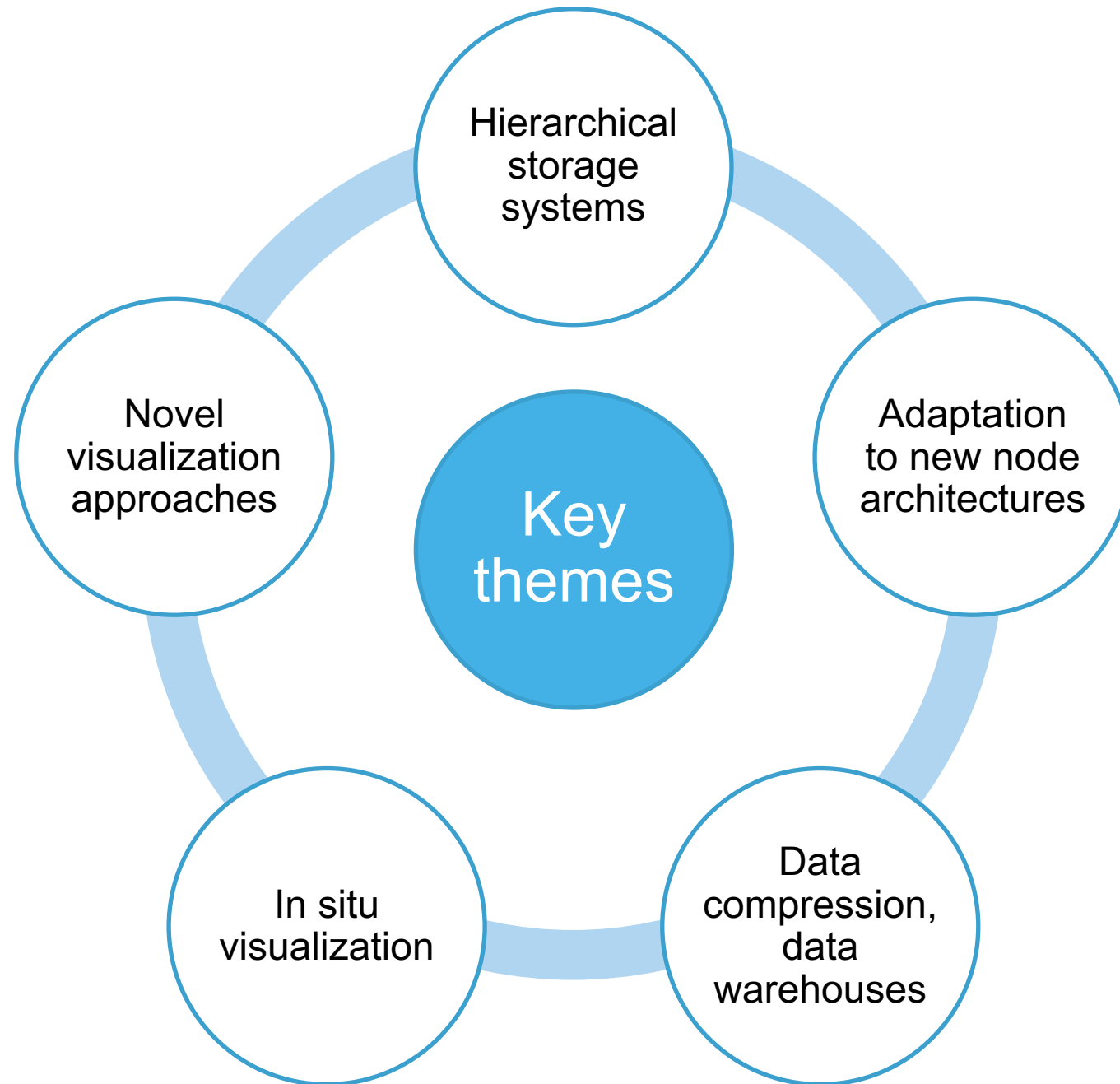
Development Tools



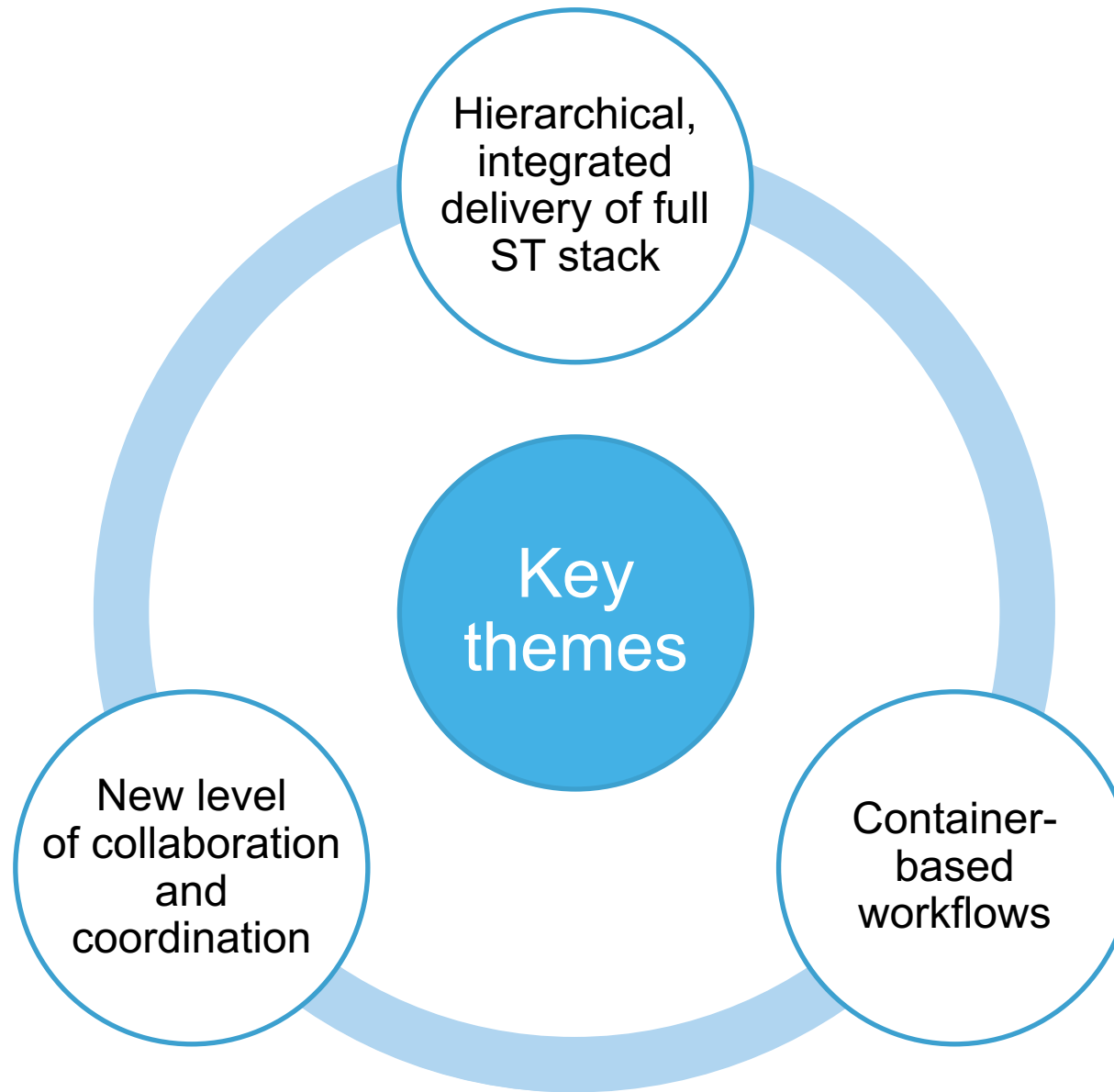
Math Libraries



Data and Visualization



Software Ecosystem and Delivery



Some Recent Application Examples of the Issues.

Simultaneous heterogeneous execution

- HPCG on Trinity
- 9380 Haswell, 9984 KNL compute nodes.
 - Haswell
 - Processor dimensions: 27x42x17
 - Local grid dimensions: 160x160x112
 - KNL
 - Processor dimensions: 27x42x34
 - Local dimensions: 160x160x152
- HPCG result: 546 TF/s (3rd).
 - Previous 180 TF/s for Haswell only.
- Key Point: For sparse codes, it's about the memory system.

HPCG on SIERRA (Power9's + 4 Voltas):

- About 10% of performance is from Power9's
- Summit 6 GPUs: Power9's less important.
- Both:
 - Code complexity challenging.
 - Runtime system complexity (MPI).
 - Work partitioning.

2-phase porting strategy: TaihuLight

- Management Processing Element (MPE)
 - 64-bit RISC core
 - support both user and system modes
 - 256-bit vector instructions
 - 32 KB L1 i-cache, 32 KB L1 data, both 4 way set associative.
 - 256 KB L2 cache, 8 way set associative
- Computing Processing Element (CPE)
 - 8x8 CPE mesh
 - 64-bit RISC core
 - support only user mode
 - 256-bit vector instructions
 - 64 KB Scratch Pad Mem (1 private per CPE)
 - Can be configured as explicit local mem or SW managed \$.
 - Each CPE has own 16KB direct mapped i-cache.

Initial port:

- Vanilla MPI, 1 rank per MPE
- 23.2 GF/s /core
- 4 vector FMA
- 2 pipes
- 16 Flops/cycle FMA
- Peak: 2/65 of node peak

Subsequent optimization:

- Offload any work to CPEs
- 11.6 GF/s /core
- 4 vector FMA
- 1 pipe
- 8 Flops/cycle FMA

CAM-SE to TaihuLight: 2017 Gordon Bell Finalist

- CAM-SE: Spectral Element Atmospheric dynamical core
 - Reported:
 - 754,129 SLOC.
 - 152,336 SLOC modified for TaihuLight (20%).
 - 57,709 SLOC added (8%).
 - 12+ team members.
 - Challenges:
 - Reusability of code seems low: Much of the optimization is specific to Sunway CPE processor.
 - Translation effort difficult to accomplish while still delivery science results: Disruptive.
 - Other notable example: Uintah (see Dec 2017 ASCAC talk)
 - Separation of runtime concerns seems to really help, but app-specific.

Some Observations from these Efforts

- Even the simplest simultaneous heterogeneous execution is difficult.
 - But maybe most apps won't care: Sequential heterogeneous execution may be sufficient.
 - But some probably will: Hard to support.
- MPI-backbone approach is very attractive.
 - Initial app port to host backbone, hotspot optimization.
 - Investment in portable programming expressions seems essential.
 - Separation of functionality expression and work/data mapping seems essential.

ECP ST Programming Models, Runtimes, Tools for EH

ECP ST MPI Efforts

- The “+” in MPI+X:
 - Two large MPI projects (MPICH and OpenMPI).
 - Heavy focus on MPI+X interaction models.
 - Heavy focus on runtime support:
 - Overlapping comm/comp, ID and remove bottlenecks.
 - High performance, correct execution.
- Support for the MPI backbone deployment model:
 - Network of host processors + accelerator offload.
 - Strong focus on OpenMP interaction.

Portable On-Node Programming

- OpenMP/OpenACC:
 - New features, MPI interoperability.
 - Standards Efforts.
- Kokkos, RAJA:
 - Write-once, transformable source code.
 - Compile-time polymorphic.
 - Memory-space abstraction (Kokkos).
 - Emphasis to evolve language standards.
- LLVM-based:
 - Leverage community convergence on LLVM.
 - OpenMP in LLVM
 - Flang: LLVM-based Fortran Compiler.
- Qthreads: Light-weight threading API, runtime.

Portable On-Node Programming (2)

- Code tuning:
 - ROSE-based code generation: Source to source, offline optimization.
 - Autotuning.
- Tools (several):
 - Insight for performance tuning.
 - New approaches for gathering statistics from the system.

Inter-node Programming

- Projects:
 - xGA – Next gen global arrays.
 - Legion – Data-driven tasking with programmable mapping.
 - ParSEC – lightweight tasking framework.
 - UPC++ - C++ based partitioned global address space model.
- Leverage for EH:
 - Experience, opportunities to abstract nodes.
 - But not first order priority (IMO).

Libraries

Many Library Efforts: Math, I/O, Data, Viz

- Math Libraries: Preserving APIs, but substantial rewrites underneath.
- Embedded data/viz:
 - In situ data analysis/compression: substantial algorithms change.
 - Conceptual APIs.
- Checkpoint/Restart:
 - Same API & execution model, very different implementation.
 - Data offload to NVRAM.
- Reusable libraries and components can be a growing portion of our portability strategy.

Software Ecosystem and Delivery

ECP Software Focus is Substantial, can impact EH

- Emphasis on quality software lifts all boats.
 - Goal: Developer productivity, software sustainability.
 - We are becoming better scientific software developers.
- Processes and practices for testing and integration.
 - Social and technology pathways for accelerating research to production delivery.
 - One major legacy of ECP.
- Software distribution channels:
 - OpenHPC (or similar) software package distribution.
 - Containers – Lower barrier to software access, in line with broader SW community.

What is Challenging for ECP to Support toward EH

- Timeline (further reduced from 10 to 7 years) leave some efforts out of scope.
- Hard to support R&D with no significant product delivery in 2021-2022 time frame.
- ECP apps and software are focused on Exascale systems (obvious).
- Off limits: Post-Exascale requirements that are not well aligned with Exascale.
- Opportunity:
 - Align Exascale and post-Exascale requirement to maximize leverage.
 - Open to ideas that make alignment better.

ECP for EH: Summary

- ECP efforts for “+” in MPI+X should aid EH significantly.
- ECP efforts for portable, efficient application development (especially on-node) should lead to apps that are easier to port to any new architectures.
- Greater emphasis on reusable, compatible software libraries and components will improve effective encapsulation of some heterogeneity.
- Providing incentives and opportunities to do a better job of scientific software development should make our community more effective and efficient.
- Help us optimize ECP alignment with post-Exascale requirements.